eMontage: An Architecture for Rapid Integration of Situational Awareness Data at the Edge

Soumya Simanta
Gene Cahill
Ed Morris

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **MAY 2013**	2. REPORT TYPE	3. DATES COVERED **00-00-2013 to 00-00-2013**
4. TITLE AND SUBTITLE **eMontage: An Architecture for Rapid Integration of Situational Awareness Data at the Edge**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Carnegie Mellon University ,Software Engineering Institute,Pittsburgh,PA,15213**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR’S ACRONYM(S)
		11. SPONSOR/MONITOR’S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**		
13. SUPPLEMENTARY NOTES **Software Engineering Institute (SEI) Architecture Technology User Network (SATURN) Conference, 29 Apr ? 3 May 2013, Minneapolis, MN.**		
14. ABSTRACT		
15. SUBJECT TERMS		

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**23**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

Motivation – Situational Awareness

First responders and others operating in the “last mile” of crisis and hostile environments are already making use of handheld mobile devices in the field to support their missions.

Rapid Incorporation of New Data Sources

- Many data sources (real-time, historical, …)
- Data is fragmented across different apps on the mobile device

Minimized Information Overload

- Edge users are under high cognitive load
- Information required is a function of user’s context and therefore dynamic

Simple Use

- Users are under high stress
- Small screen devices

Resource Constrained Hostile Environment

Hostile Environments Characteristics

Wimpy edge nodes

- Limited resources (CPU, battery and memory) on mobile nodes
- Example: Expensive computations on a smartphone may drain the battery fast

Limited or no end-to-end network connectivity

- Implicit assumption of WAN connectivity is not always valid
- Example: No access to internet during a disaster, DoS attack

High cognitive load

- Application latency and fidelity become important
- Example: A slow application will increase the cognitive load on the user

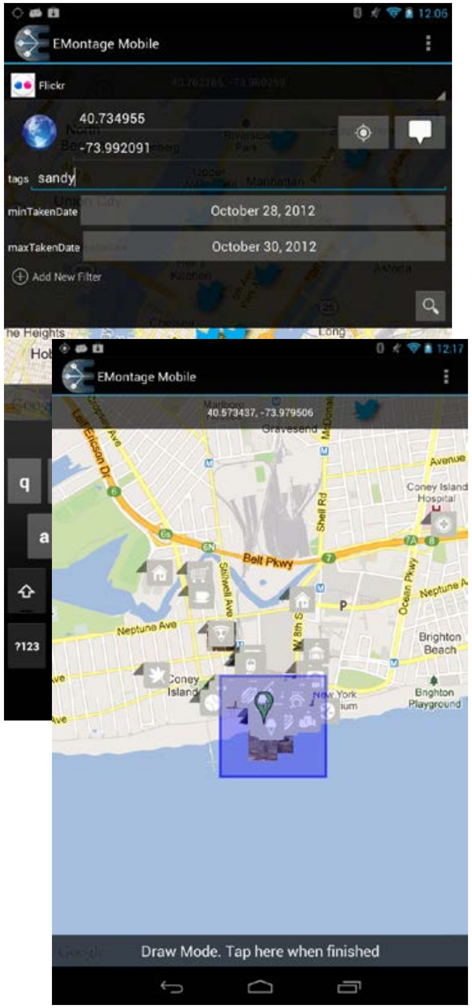
Bounded elasticity

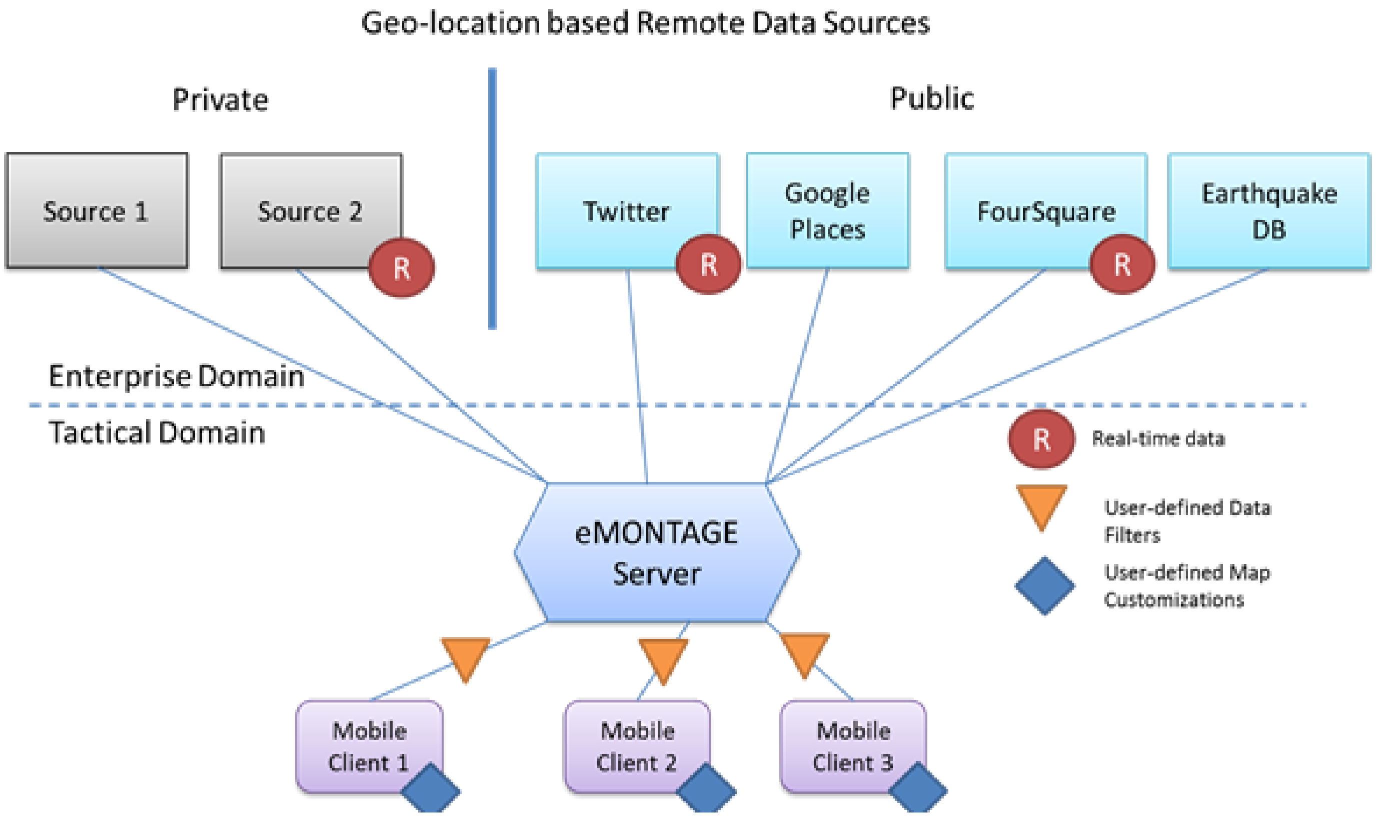
- Upper bound on number of consumers known in advance
- Example: Fixed number of first responders in a location

Dynamic environment

- Static deployment topologies cannot be assumed; Survivability essential
- Example: An automobile with a server may not be available

Context Diagram





Architecturally Significant Requirements

Extensibility

- Add new data source quickly with minimal impact on existing sources

Runtime Configurability

- Make data sources user configurable (e.g., using data filters) at runtime

Performance

- Minimize network bandwidth usage of the tactical network

Energy Efficiency

- Optimize energy consumption on mobile handheld devices

Usability

- Provide a responsive and unified user interface

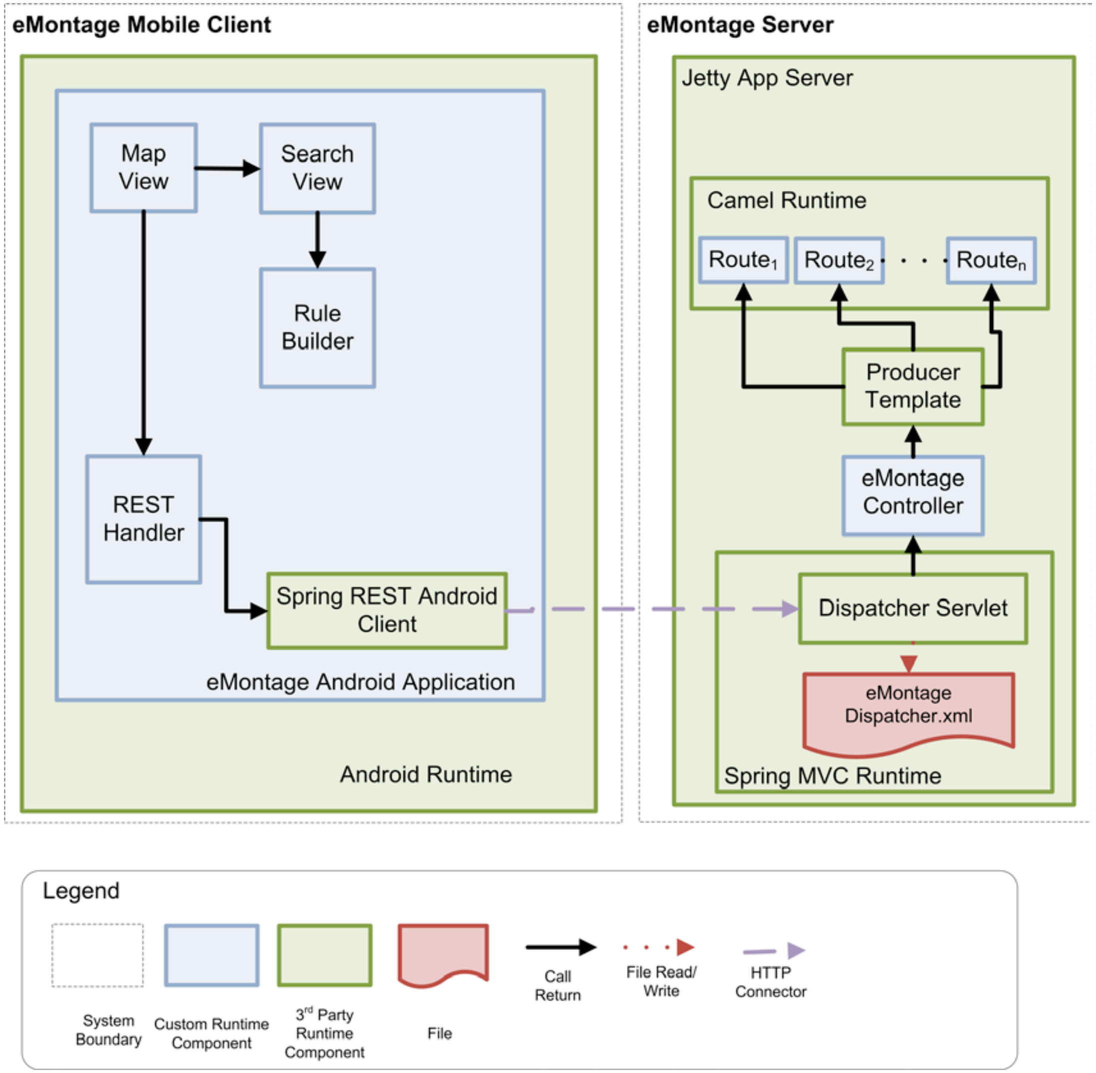
Availability

- Support intermediate disconnections with remote data sources

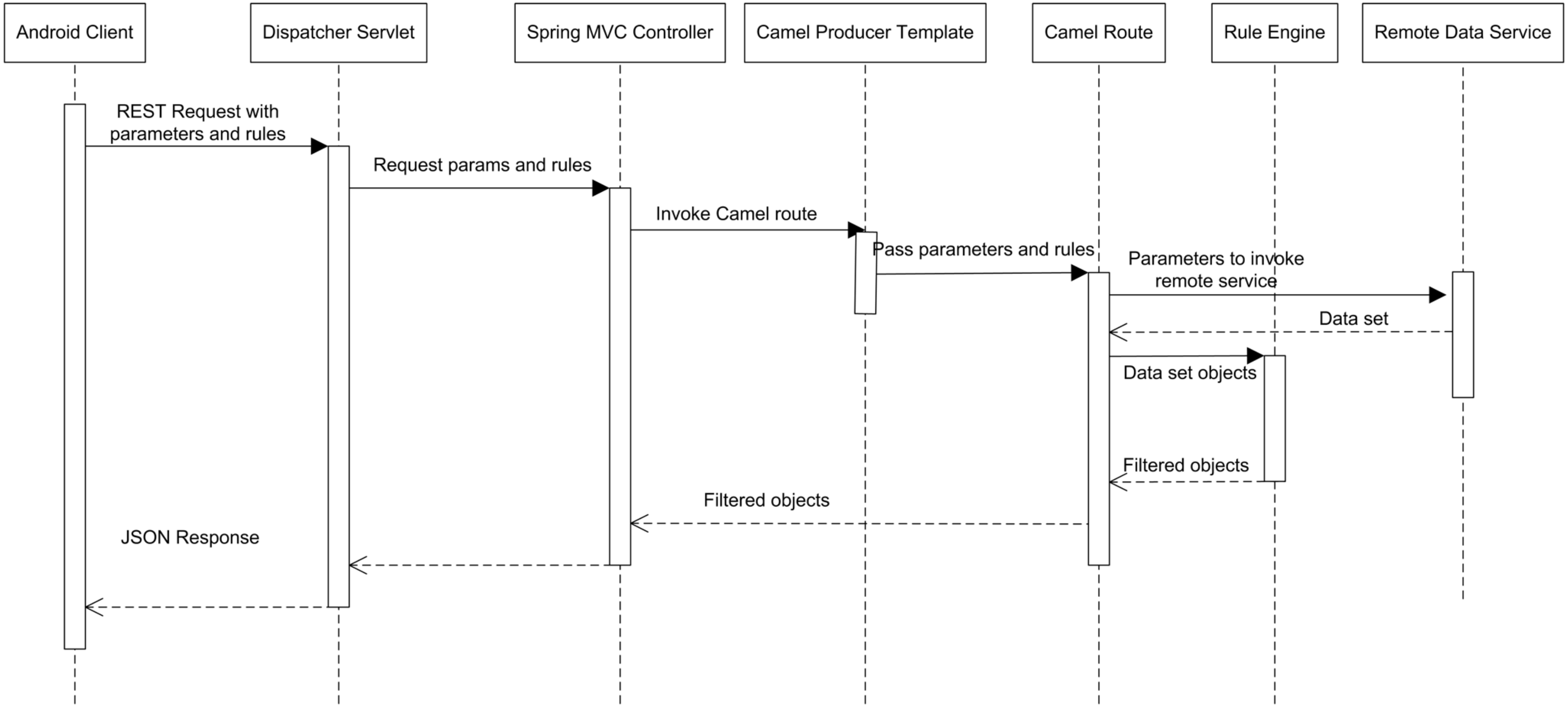
Security

- Support existing security protocols and provide transport layer security

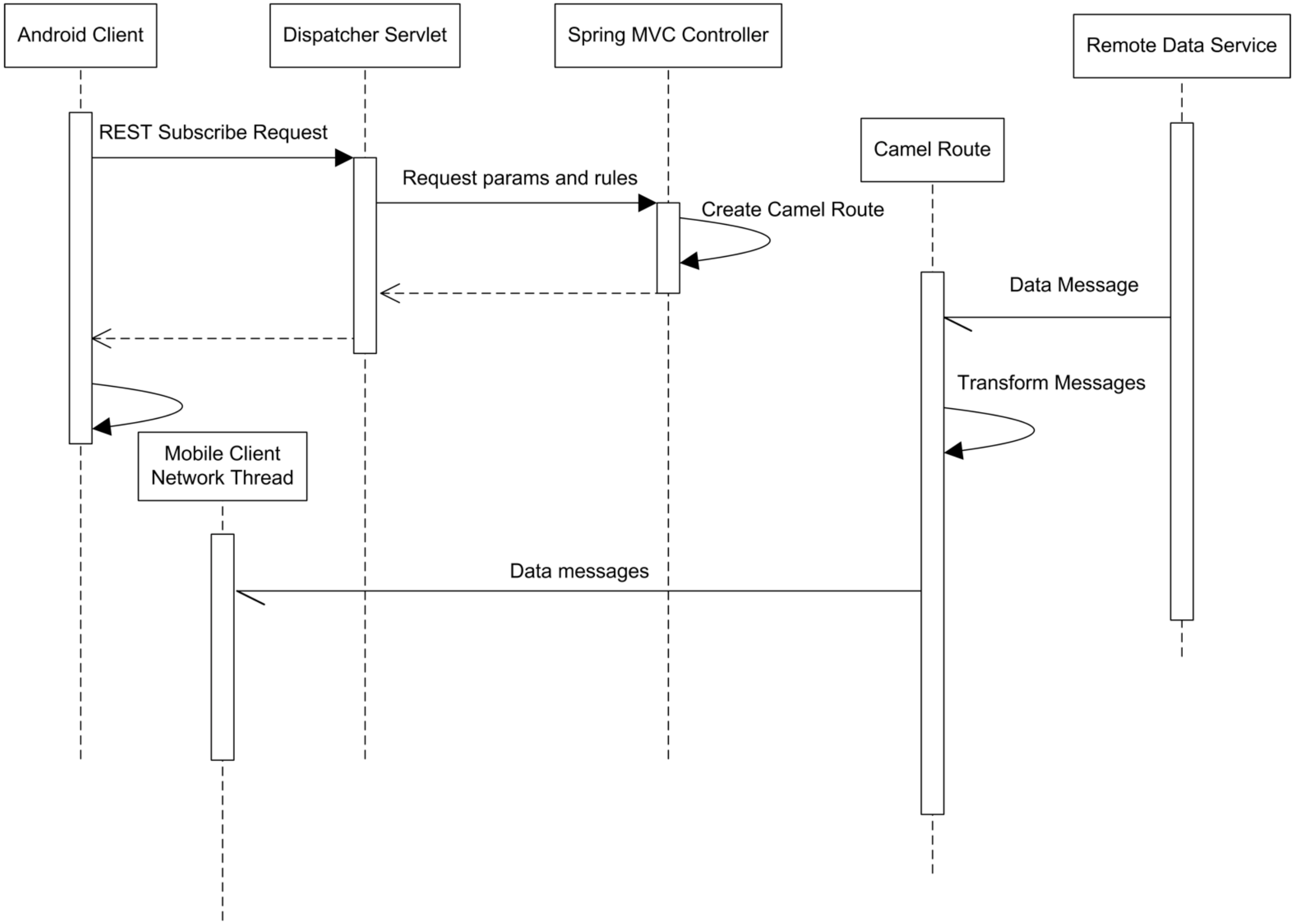
Runtime C&C View



Request Response Interaction

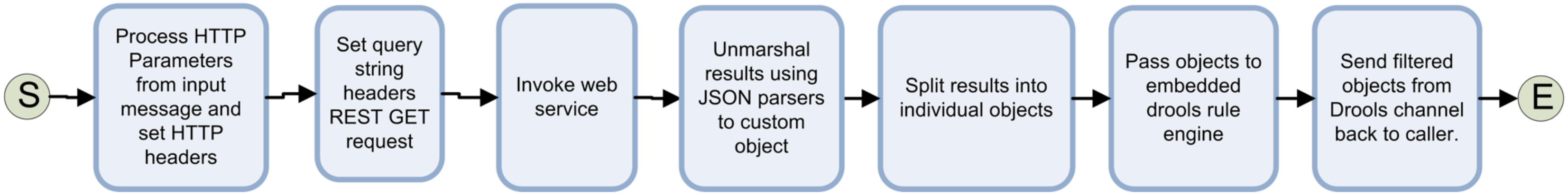


Publish Subscribe Interaction

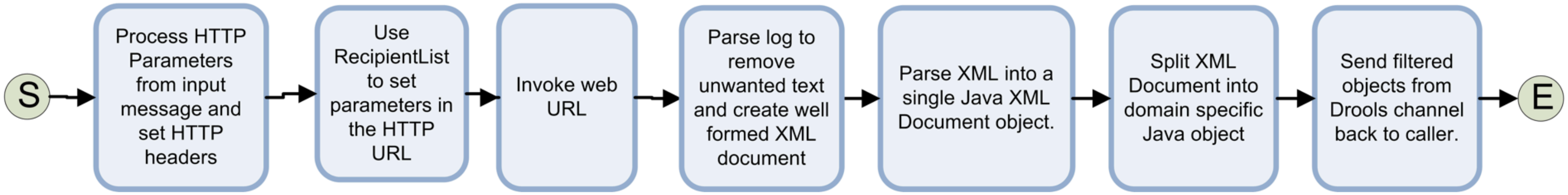


Example Routes

Google Place Route



National Weather Service Alerts Route



Extensibility – Adding New Data Sources

Problem
- New data sources are available in the field
- Adding and validating them is time consuming

Assumptions
- The data source has a remote API
- The data format is defined and stable

Solution
- Minimize coupling between data sources by encapsulating each data source
- Implement common connectors (request/response and publish subscribe)

Future extensions

Automate common tactical integration patterns to provide an end-user programing interface

Data Model

Data model is represented as objects (POJOs) shared between clients and server

Data model changes must be synchronized between clients and servers

- Our assumption: data model is “relatively” stable

Data model can be created in the following ways

- **Manual definition** – works best in case of a simple data model
- **Code generation** – WSDL2Java to generate code
- **Reuse existing library** – Twitter4J is an existing Java implementation of Twitter API

Mashup Mechanism

Merging data across data models is a mechanism to relate data across models.

- Example: foreign keys in a relational database

In eMontage, we assume a large proportion of situational awareness data has some form of geo-location associated with it

- use geo-location as the common key

All data is currently mashed up on a map-based interface.

- as long as two data elements from different data sources are referenced by geo-location (latitude and longitude), they will always be displayed correctly on a map
- the actual relating of information will happen with the user

Example Data Sources

Data Source Name	Data Format	Wire Protocol	Security Mechanisms	Data Model Complexity	User Interface Complexity
Google Places	JSON	REST	Token-based	Low	Geo-points on map
Twitter	JSON	REST	Token-based	High	Geo-points on map
FourSquare	JSON	REST	Token-based	High	Geo-points on map
Private Data source 1 (real-time)	XML	UDP	No security mechanism	Low	Polygons, Geo-points on map
Private data source 2 (historical)	SOAP	HTTP	Custom	High	Geo-points on map
National Weather Service Alerts	Custom log files	HTTP	No security mechanism	Medium	Polygons

Configurability- User-defined Runtime Filtering

Problem	Information overload
Assumption	The user knows what information they need in a particular context (e.g., location, keywords, date ranges)
Solution	Provide mechanisms that allow users to reduce the volume of information • using rule-based filtering at runtime
Future extensions	Provide “data discovery” mechanisms (e.g., visualizations, clusters, outliers) when the user does not know what information they need

Usability - Unified User Interface

Problem	Data is fragmented across multiple applications and databases
Assumption	• Data is geo-coded • A unified view provides more value compared to isolated views of data from different sources
Solution	Mashup of geo-code data viewed on a map allows *visual* unification of data
Future extensions	• Provide other non-map based visualizations • Provide data join mechanisms

Performance - Minimized Bandwidth Utilization

Problem	Bandwidth is a scare resource at the "last mile" of edge
Assumption	Possible to have an intermediary node in the network
Solution	Add an intermediary node • Use filtering at the source (only send information is required by the mobile nodes) • Transform to a more bandwidth optimized format (e.g., XML to JSON/Protocol Buffer)
Future extensions	Use protocol transformation (use a SPDY instead of HTTP)

Power Consumption - Offloading Expensive Computation

Problem	Mobile nodes have limited resources (CPU, battery and memory)
Assumption	Possible to have an intermediary node in the network
Solution	Perform expensive computation (e.g., XML parsing, multiple network calls) on a proximate, relatively resource rich node
Future extensions	Use multi-node cloudlets to increase performance and fault tolerance

Availability - Disconnected Operations

Problem	Edge nodes may have to work in disconnected or semi-connected mode (from enterprise/TOC network)
Assumption	• Possible to deploy a resource-rich node locally (e.g., on an automobile) • Real-time data is generated *locally* • Possible to know in advance what data will be required for a mission (e.g., maps by locations)
Solution	Localize and cache data sources on a cloudlet.
Future extensions	Use persistent distributed caching. Adaptive pre-fetching to support intermittent disconnections

Architectural Alternatives

Native Mobile Client Only

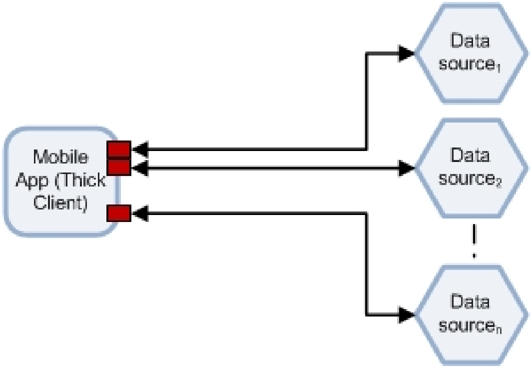
- A **native**, **mobile client** app directly connected to backend data sources

Mobile Browser-Server

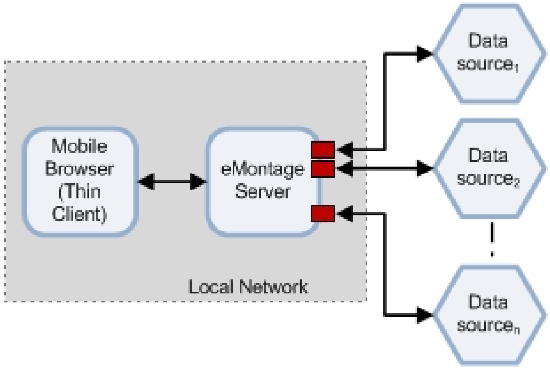
- A **mobile browser** client to a server that acts as an **intermediary** between the mobile client and the backend data sources

Native Mobile Client-Server

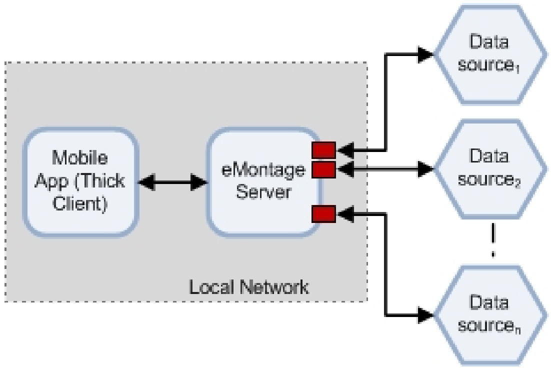
- A **native**, **mobile client** app connected to an in **intermediary** server



Native Mobile Client Only



Mobile Browser-Server



Native Mobile Client-Server

Architectural Alternatives

	Native Mobile Client Only	Mobile Browser-Server	Native Mobile Client-Server
Reuse of COTS	Low	High	High
Protocol/Data format transformation	No	Yes	Maybe
Intermediate filtering	No	Yes	Yes
Bandwidth optimization	No	Yes	Maybe
Disconnected operations	No	Yes	Yes
Rich user interface	Yes	Yes	Maybe
Energy efficiency	No	Yes	Maybe
Fault Tolerance	High	Low	Low
Caching	No	Yes	Yes
Runtime modifiability	Low	High	High

Current and Future Work

More intuitive user interface

- Support other views of data
- Provide data exploration and discovery capabilities

Focus on performance

- Use caching
- Allow use of multiple processors/cores when possible

Add security mechanisms

- Use with Wave Relay radios
- Add transport and message level encryption

Integrate with edge analytics

- Build edge analytics techniques on top of current eMontage implementation

Questions